3. kotitehtävän palautus

- Jokainen resoluutioaskel kohtaan 1 (yleisimmät unifioijat, muuttujien nimeäminen). Kohta 1 **ei ole** OTTER-tehtävä.
- Ratkaisuksi laadittu OTTER-syötetiedosto kohtaan 2 ja selitys jokaisesta lauseesta (samankaltaisina toistuvia lauseita ja atomilauseita ei tarvitse erikseen selittää)
- Muutama ajoesimerkki:
  - OTTERin löytämiä todistuksia (pelkkä todistus riittää)
  - kyselyitä joille OTTER ei löytänyt todistusta, ja joille ei pitänytkään todistusta löytyä (jotta näkyy, ettei "tietokanta" ole ristiriitainen).
- Palautettava viimeistään 7.12.1998.

Logiikkaohjelmat

- *Horn-klausuuli* on klausuuli, jossa on enintään yksi positiivinen literaali.

  Esim. $\{A\}$, $\{A, \neg B, \neg C\}$, $\{\neg D, \neg E, \neg F\}$.

  Prologin notaatiolla: `A.` `A :- B, C.` `:- D, E, F.`
- *Ohjelmaklausuuli* on klausuuli, jossa on täsmälleen yksi positiivinen literaali.

  Esim. $\{A\}$, $\{A, \neg B, \neg C\}$
- *Maaliklausuuli* on klausuuli jossa on vain negatiivisia literaaleja.

  Esim. $\{\neg D, \neg E, \neg F\}$
- *Prolog-ohjelma* on joukko ohjelmaklausuuleja.

LI/LD/SLD resoluutio

L=Linear

I=Input

D=Definite

S=Selection Function

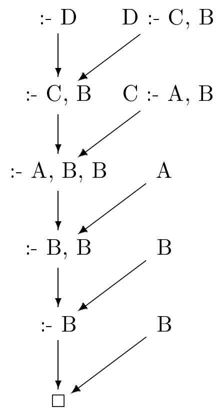
`A.`

`B.`

`C :- A, B.`

`D :- C, B.`

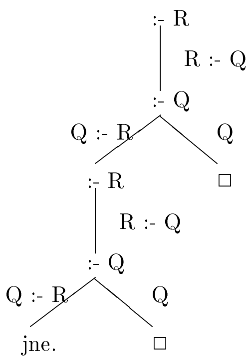
Kysely `:- D`



Väite. Jos S on toteutumaton Prolog-ohjelma (joukko ohjelmaklausuuleja + maaliklausuuli), niin S:lle on olemassa SLD-refutaatio.

- Tämä täydellisyysominaisuus menetetään Prologissa, jos käytetään syvyyshakua (kts. seuraava esimerkki).

Prologin hakuproseduurin epätäydellisyys



R :- Q.
Q :- R.
Q.

Kyselynä :- R.

Prolog-ohjelmat predikaattilogiikassa

Prolog	Klausuuliesitys	Vastaava lause
`p(X)`	$\{P(x)\}$	$\forall x(P(x))$
`r(X,Y) :- q(X)`	$\{R(x,y), \neg Q(x)\}$	$\forall x \forall y(Q(x) \rightarrow R(x,y))$
`:-q(X),s(X)`	$\{\neg Q(x), \neg S(x)\}$	$\neg \exists x(Q(x) \wedge S(x))$

- Ohjelmaklausuulien muuttujat ovat universaalisti kvantifioidut ja kyselyn muuttujat ovat eksistentiaalisesti kvantifioidut.

Listanotaatio:

Prolog	Vastaava termi
`[]`	e
`[X|[Y|Z]]`	$\mathrm{c}(x, \mathrm{c}(y, z))$
`[X,Y,Z]`	$\mathrm{c}(x, \mathrm{c}(y, \mathrm{c}(z, e)))$

Lineaarinen input-resoluutio

Olkoon P joukko klausuuleja ja G kysely (sisältää vain negatiivisia literaaleja). Allaoleva on refutaatio *lineaarisella input-resoluutiolla* (LI), joss $G_0 = G$ ja kukin C_i on P:n klausuuli, missä muuttujat on tarvittaessa uudelleennimetty.

$$\begin{tikzcd} C_0 \arrow[dr] & C_1 \arrow[dr] & C_2 \arrow[dr] & & \dots\ C_n \arrow[dr] & \\ G_0 \arrow[r] & G_1 \arrow[r] & G_2 \arrow[r] & G_3 & \dots\ G_n \arrow[r] & \square \end{tikzcd}$$

Väite. (Virheettömyys ja täydellisyys) Olkoon P joukko ohjelmaklausuuleja ja G kysely. Nyt $P \cup \{G\}$ on toteutumaton, jos ja vain jos on olemassa $P \cup \{G\}$:n refutaatio lineaarisella input-resoluutiolla siten, että refutaatio alkaa klausuulista $G_0 = G$.

Todistus: (Virheettömyys) Resoluutioaskeleet säilyttävät toteutuvuuden. (Täydellisyys) Lineaarinen resoluutio takaa refutaation olemassaolon (T II.14.4). Klausuuleissa G_i on vain negatiivisia literaaleja, joten jokaisella askeleella klausuuli C_i on välttämättä ohjelmaklausuuli (jossa on siis täsmälleen yksi positiivinen literaali).

Järjestetyt Horn-klausuulit

Olkoot $G = \{\neg A_0, \dots, \neg A_n\}$ ja $C = \{B, \neg B_0, \dots, \neg B_m\}$ järjestettyjä klausuuleita, ja θ literaalien A_i ja B yleisin unifioija. Klausuuleista G ja C saadaan resoluutiolla järjestetty klausuuli $G = \{\neg A_0, \dots, \neg A_{i-1}, \neg B_0, \dots, \neg B_m, \neg A_{i+1}, \dots, \neg A_n\}\theta$.

Olkoon P joukko järjestettyjä klausuuleita ja G järjestetty kysely. Allaoleva on refutaatio *LD-resoluutiolla* (Linear Definite), jos $G_0 = G$ ja kukin C_i on P:n klausuuli, missä muuttujat on tarvittaessa uudelleennimetty.

$$
\begin{tikzcd}
C_0 \arrow[dr] & C_1 \arrow[dr] & C_2 \arrow[dr] & \dots C_n \arrow[dr] & \\
G_0 \arrow[r] & G_1 \arrow[r] & G_2 \arrow[r] & G_3 \dots G_n \arrow[r] & \square
\end{tikzcd}
$$

LD-resoluutio valintasäännöllä = SLD-resoluutio

Valintasääntö on funktio R, joka valitsee järjestetystä ei-tyhjästä klausuulista C literaalin $L = R(C)$.

Prologin valintasääntö R_P:

Klausuulille $C = \{\neg A_1, \dots, \neg A_n\}$, $R_P(C) = \neg A_1$.

Olkoon P joukko järjestettyjä klausuuleita ja G järjestetty kysely. Allaoleva on refutaatio *SLD-resoluutiolla* (Linear Definite), jos $G_0 = G$ ja kukin C_i on P:n klausuuli (muuttujat uudelleennimetty), ja *askeleella i resoluutiosääntöä sovellettiin klausuuleihin G_i ja C_i literaaliin* $R(G_i)$.

$$
\begin{tikzcd}
C_0 \arrow[dr] & C_1 \arrow[dr] & C_2 \arrow[dr] & \dots C_n \arrow[dr] & \\
G_0 \arrow[r] & G_1 \arrow[r] & G_2 \arrow[r] & G_3 \dots G_n \arrow[r] & \square
\end{tikzcd}
$$

Tämä on edelleen virheetön ja täydellinen resoluutiomenetelmä (suhteessa $P \cup \{G\}$ toteutumattomuuteen).

Prologin hakumekanismi

```
r(X) :- q(X,b).
r(a) :- s(a,Y).
r(X) :- p(X).
...
```

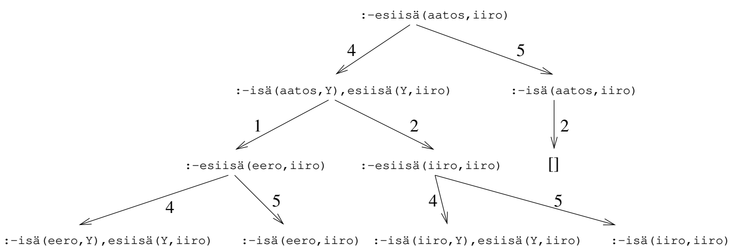
Kysely: `:-r(Z)`

$$
\begin{tikzcd}
\texttt{:-r(Z)} \arrow[d] & \texttt{r(X):-q(X,b)} \arrow[dl] & \texttt{:-r(Z)} \arrow[d] & \texttt{r(a):-s(a,Y)} \arrow[dl] & \texttt{:-r(Z)} \arrow[d] & \texttt{r(X):-p(X)} \arrow[dl] \\
\texttt{:-q(Z,b)} \arrow[d] & C_1 \arrow[dl] & \texttt{:-s(a,Y)} \arrow[d] & C_1 \arrow[dl] & \texttt{:-p(Z)} \arrow[d] & C_1 \arrow[dl] \\
G_2 & & G_2 & & G_2 & \\
\vdots & & \vdots & & \vdots &
\end{tikzcd}
$$

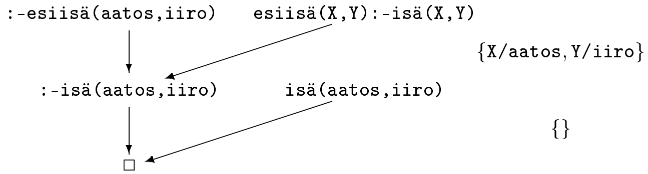
Esimerkki

```
isä(aatos,eero).
isä(aatos,iiro).
isä(oiva,aatos).

esiisä(X,Z) :- isä(X,Y), esiisä(Y,Z).
esiisä(X,Y) :- isä(X,Y).

isoisä(X,Z) :- isä(X,Y), isä(Y,Z).
```

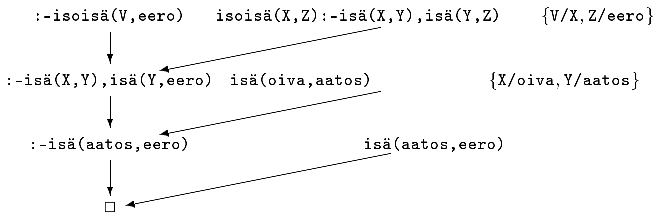

Prologin hakupuu



Tuloksena saatava lineaarinen resoluutiotodistus



Vastauksena sidokset kyselyn muuttujille



Unifioijien kompositio: {V/oiva, X/oiva, Z/eero, Y/aatos}

Esimerkki

$$T\forall x \forall y \mathrm{Me}(x, \mathrm{c}(x,y))$$
$$T\forall x \forall y \forall z (\mathrm{Me}(x,z) \rightarrow \mathrm{Me}(x, \mathrm{c}(y,z)))$$
$$F\mathrm{Me}(a, \mathrm{c}(b, \mathrm{c}(a,e)))$$
$$T\mathrm{Me}(a, \mathrm{c}(a,e)) \rightarrow \mathrm{Me}(a, \mathrm{c}(b, \mathrm{c}(a,e)))$$

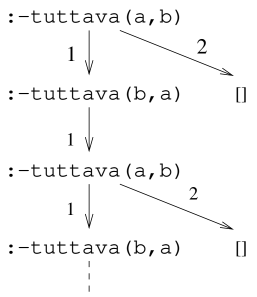
$$F\mathrm{Me}(a, \mathrm{c}(a,e)) \qquad\qquad T\mathrm{Me}(a, \mathrm{c}(b, \mathrm{c}(a,e)))$$
$$T\mathrm{Me}(a, \mathrm{c}(a,e)) \qquad\qquad \otimes$$
$$\otimes$$

`Me(X,[X|Y]).    Me(X,[Y|Z]) :- Me(X,Z).    :- Me(a,[b,a])`

Syvyyshaku: epätäydellisyys

1: `tuttava(X,Y) :- tuttava(Y,X)`   Kysely: `:-tuttava(a,b)`

2: `tuttava(a,b)`



Prologin not-operaattori

- Halutaan käyttää muunkinlaisia sääntöjä kuin $P(t_1, \ldots, t_n)$ ja $P(t_1, \ldots, t_n)$:- $P_1(s_{11}, \ldots, s_{1n_1}), \ldots, P_m(s_{m1}, \ldots, s_{mn_m})$. Esim. ei-operaatiota säännön ehto-osassa.
- Koko predikaattilogiikka liian monimutkainen: ohjelmien suorittaminen ei olisi kyllin tehokasta
- Laajennetaan resoluutioproseduuria:
  - Käytetään predikaattilogiikan negaation "¬" asemesta toisentyyppistä negaatiota "`not`".
  - Tulokset käytännön sovelluksissa useimmiten haluttuja

NOT – negaatio epäonnistumisena

"Jos ei voida päätellä ϕ, päätellään `not` ϕ"

```
koululainen(oskari).
koululainen(emilia).
sairas(oskari).
koulussa(X) :- koululainen(X), not sairas(X).
```

1. Kysely: `:-koulussa(oskari)`
2. Kysely: `:-koulussa(emilia)`

Ero klassiseen negaatioon

```
nuori(a).
vanha(X) :- not nuori(X).
```

Prolog vastaa kyselyyn vanha(b) myönteisesti. Lause vanha(b) ei ole kuitenkaan ohjelman looginen seuraus, jos klausuuleissa esiintyvä Prologin negaatio tulkitaan predikaattilogiikan negaatioksi:

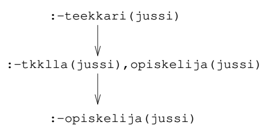
$$
\begin{array}{c}
T\text{nuori}(a) \\
T\forall x(\neg\text{nuori}(x) \to \text{vanha}(x)) \\
F\text{vanha}(b) \\
T(\neg\text{nuori}(b) \to \text{vanha}(b))
\end{array}
$$

$$
\begin{array}{ccc}
F\neg\text{nuori}(b) & \qquad\qquad & T\text{vanha}(b) \\
T\text{nuori}(b) & & \otimes
\end{array}
$$

Määritelmä. *Äärellinen epäonnistunut SLD-puu* P:lle G:lle ja R:lle on äärellinen SLD-resoluution hakupuu, jonka mikään lehti ei ole tyhjä klausuuli $\square$.

Äärellinen epäonnistunut SLD-puu kyselylle `teekkari(jussi)`:

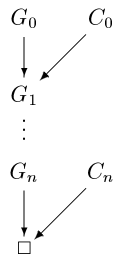
```
teekkari(X) :- tkklla(X),
               opiskelija(X).
opiskelija(teemu).
tkklla(jussi).
tkklla(teemu).
```



SLDNF-resoluutio

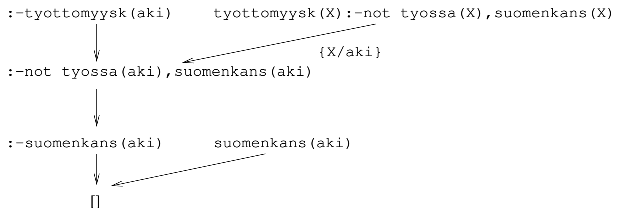
Olkoon P joukko järjestettyjä klausuuleita ja G järjestetty kysely. Vieressä oleva on refutaatio *SLDNF-resoluutiolla* (Selection function, Linear Definite, Negation as Failure), jos $G_0 = G$ ja kukin C_i on P:n klausuuli (mahdollisesti muuttujat uudelleennimettynä), ja

- $R(G_i)$ on not L, L on muuttujaton, ja L:llä P:llä ja R:llä on äärellinen epäonnistunut SLDNF-puu ja G_{i+1} on $G_i - \{\text{not } L\}$, tai
- askeleella i suoritettiin resoluutio klausuuleille G_i ja C_i literaalilla $R(G_i)$.




```
suomenkans(aki).
suomenkans(eve).
tyossa(eve).
tyottomyysk(X) :- not tyossa(X), suomenkans(X).
```

Kyselylle `:-tyottomyysk(aki)` saadaan seuraava SLDNF-refutaatio:



Huomaa, että alikysely `:-tyossa(aki)` epäonnistuu.

Prolog-ohjelman täydentäminen

$$P(t_{11}, \ldots, t_{1n}) \text{ :- } l_{11}, \ldots, l_{1m_1}$$
$$\vdots$$
$$P(t_{r1}, \ldots, t_{rn}) \text{ :- } l_{r1}, \ldots, l_{rm_r}$$

P-predikaatin klausuulit täydenetään lauseeksi

$$\forall x_1 \cdots \forall x_n \ (P(x_1, \ldots, x_n) \leftrightarrow$$
$$(\exists y_{11} \cdots \exists y_{1v_1} (x_1 = t_{11} \wedge \cdots \wedge x_n = t_{1n} \wedge l_{11} \wedge \cdots \wedge l_{1m_1}) \vee$$
$$\ldots$$
$$\vee \exists y_{r1} \cdots \exists y_{rv_r} (x_1 = t_{r1} \wedge \cdots \wedge x_n = t_{rn} \wedge l_{r1} \wedge \cdots \wedge l_{rm_r})))$$

Ohjelmaa P vastaava lausejoukko Comp(P) sisältää täydennettyjen klausuulien lisäksi yksikäsitteisten nimien aksioomat.

Ohjelma

```
suomenkansalainen(aki).
suomenkansalainen(eve).
tyossa(eve).
tyottomyyskorvausta(X) :- not tyossa(X),
                          suomenkansalainen(X).
```

on täydennettynä

$\forall x$ (suomenkansalainen$(x) \leftrightarrow (x =$ aki $\vee\ x =$ eve))

$\forall x$ (tyossa$(x) \leftrightarrow (x =$ eve))

$\forall x$ (tyottomyyskorvausta$(x) \leftrightarrow$

$(\neg$ tyossa$(x) \wedge$ suomenkansalainen$(x)))$

$\neg$(aki $=$ eve)

Väite. Jos not-operaatiota sisältävällä ohjelmalla P ja kyselyllä G $= \{\neg L_1, \ldots, \neg L_m\}$ on SLDNF-refutaatio, niin

$$\text{Comp(P)} \models \exists x_1 \cdots \exists x_n (L_1 \wedge \cdots \wedge L_m).$$

Esimerkki. Kaikkia Comp(P):n loogisia seurauksia ei saada todistetuksi SLDNF-resoluutiolla:

```
v(X) :- not q(X).
q(a) :- not r(a).
r(a) :- p(a).
r(a) :- not p(a).
p(X) :- p(f(X)).
```

$\forall x(v(x) \leftrightarrow \neg q(x))$

$q(a) \leftrightarrow \neg r(a)$

$r(a) \leftrightarrow (p(a) \vee \neg p(a))$ eli $r(a)$

$\forall x(p(x) \leftrightarrow p(f(x)))$

Kyselyyn `:-v(a)` ei saada vastausta, vaikka Comp(P) $\models v(a)$.

Jos negaation yhteydessä käytetään muuttujia sisältäviä literaaleja, SDLNF-resoluution virheettömyys menetetään (floundering):

```
koululainen(oskari).
sairas(liisa).
koulussa(X) :- not sairas(X), koululainen(X).
```

Kysely `:-koulussa(X)` epäonnnistuu.

Comp(P) = { $\forall x$(koululainen$(x) \leftrightarrow x =$ oskari),
$\forall x$(sairas$(x) \leftrightarrow x =$ liisa),
$\forall x$(koulussa$(x) \leftrightarrow (\neg$sairas$(x) \wedge$ koululainen$(x))$),
$\neg$(oskari $=$ liisa)}

Mutta: Comp(P) $\models \exists x$koulussa(x).